(12) UK Patent Application (19) GB (11) 2 320 111 (13) A

(43) Date of A Publication 10.06.1998

(21) Application No 9625443.8

(22) Date of Filing 06.12.1996

(71) Applicant(s)
JBA Holdings plc

(Incorporated in the United Kingdom)

4 Wheeleys Road, Edgbaston, BIRMINGHAM, B15 2LD, United Kingdom

(72) Inventor(s)
Christopher George Cowan
Nick John Gilbert
William Graham Lyttle
Gary Charles Lewis

(74) Agent and/or Address for Service
Dibb Lupton Alsop
Fountain Precinct, Balm Green, SHEFFIELD, S1 1RZ, United Kingdom

(51) INT CL^6
G06F 9/44

(52) UK CL (Edition P)
G4A APL

(56) Documents Cited
GB 2266982 A GB 2077966 A EP 0710909 A1
WO 88/07719 A2 US 5386568 A
WordPerfect 6.0a, 20 April 1994, WordPerfect Corporation

(58) Field of Search
UK CL (Edition O) G4A APL APX
INT CL^6 G06F 9/44 9/46

(54) **Data processing system and method for software development/configuration**

(57) The present invention finds particular application in the design of scalable business systems in which a plurality of program code entities or objects are progressively constructed to computerise a business process, and overcomes problems when software has to be re-developed, in particular, when the order of execution of objects implementing various functions has to be changed. The present invention provides a data processing system comprising memory 212 for storing a plurality of executable entities, means for establishing an order of execution of selectable ones of the executable entities, and means 410 for executing the selected executable entities according to that order without modification of the selected executable entities. The entities may be objects, functions, self-contained programs, or DLL tasks. A drag-and-drop method, using a graphical user interface 400, may be used to establish the order. The entities may have shared access to a common data structure and may be executed according to a timed schedule.

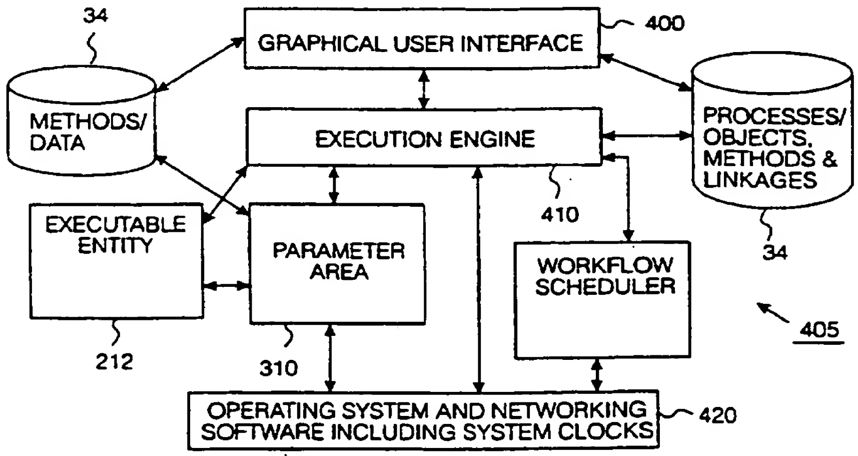
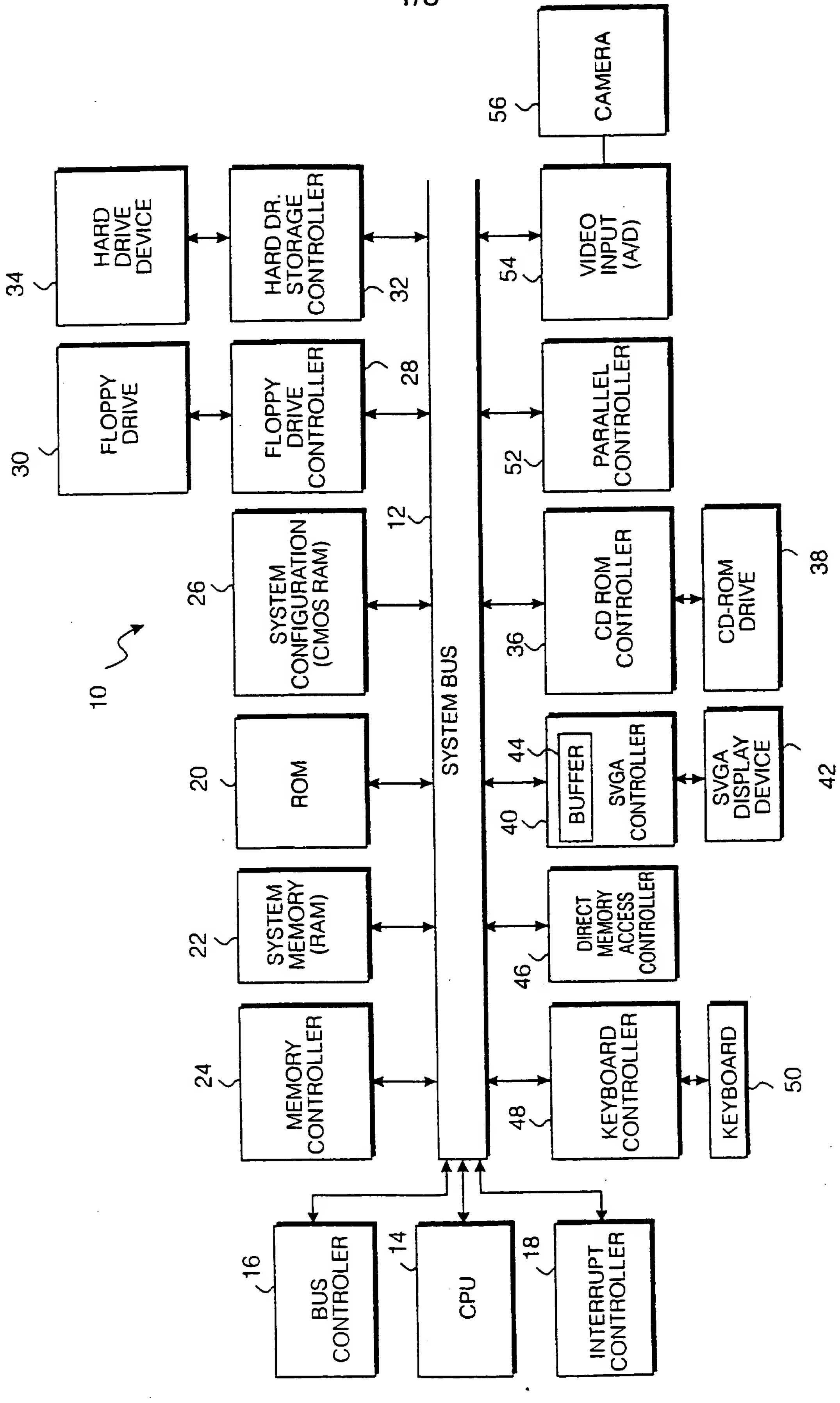


Fig. 4

GB 2 320 111 A

At least one drawing originally filed was informal and the print reproduced here is taken from a later filed formal copy.



10
CAMERA
56
HARD DRIVE DEVICE
34
HARD DR. STORAGE CONTROLLER
32
VIDEO INPUT (A/D)
54
FLOPPY DRIVE
30
FLOPPY DRIVE CONTROLLER
28
PARALLEL CONTROLLER
52
SYSTEM CONFIGURATION (CMOS RAM)
26
12
CD ROM CONTROLLER
36
CD-ROM DRIVE
38
ROM
20
SYSTEM BUS
BUFFER
44
SVGA CONTROLLER
40
SVGA DISPLAY DEVICE
42
SYSTEM MEMORY (RAM)
22
DIRECT MEMORY ACCESS CONTROLLER
46
MEMORY CONTROLLER
24
KEYBOARD CONTROLLER
48
KEYBOARD
50
BUS CONTROLER
16
CPU
14
INTERRUPT CONTROLLER
18

Fig. 1

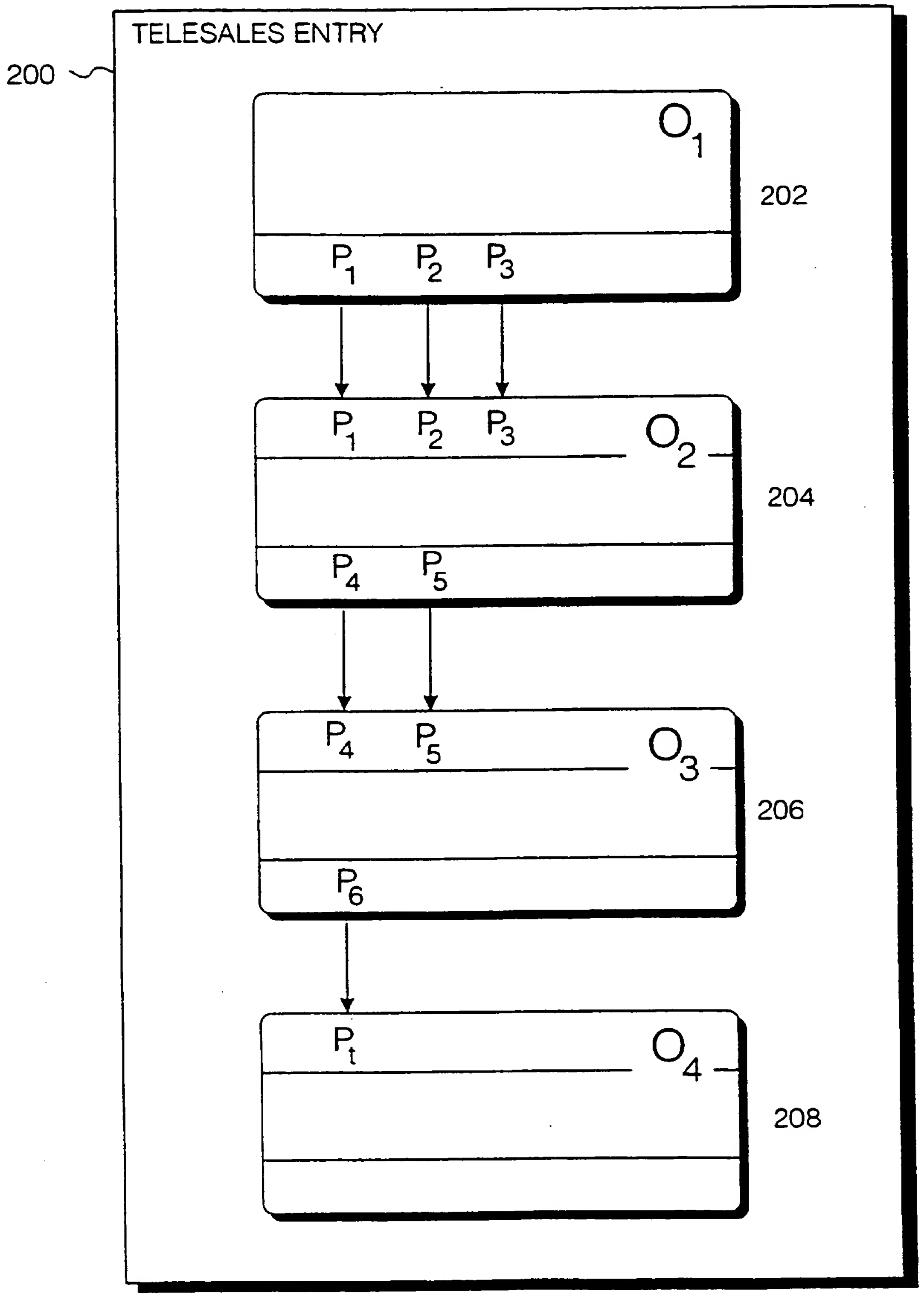

TELESALES ENTRY
200
O_1
202
P_1 P_2 P_3
P_1 P_2 P_3
O_2
204
P_4 P_5
P_4 P_5
O_3
206
P_6
P_t
O_4
208

Fig. 2a

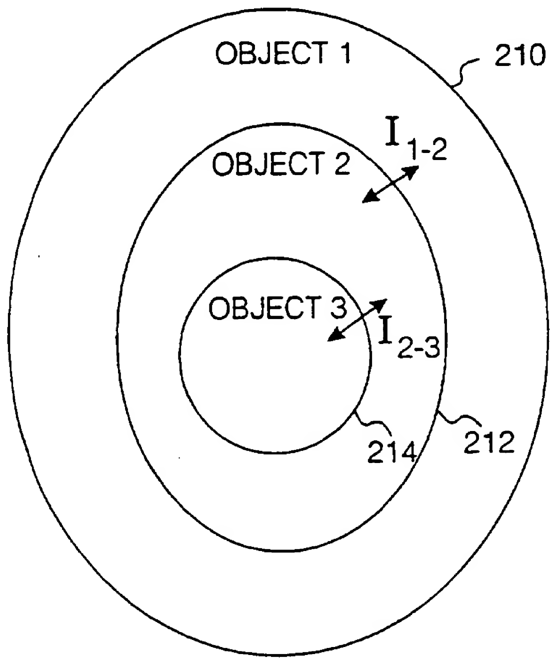

OBJECT 1
210
OBJECT 2
I_{1-2}
OBJECT 3
I_{2-3}
214
212

Fig. 2b

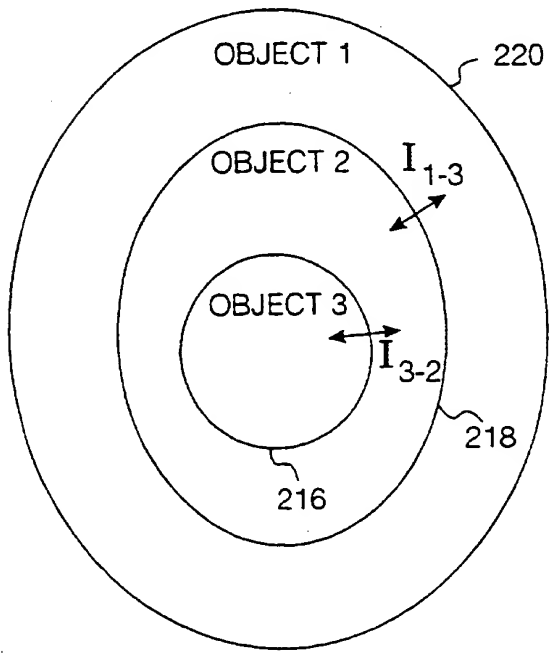

OBJECT 1
220
OBJECT 2
I_{1-3}
OBJECT 3
I_{3-2}
218
216

Fig. 2c

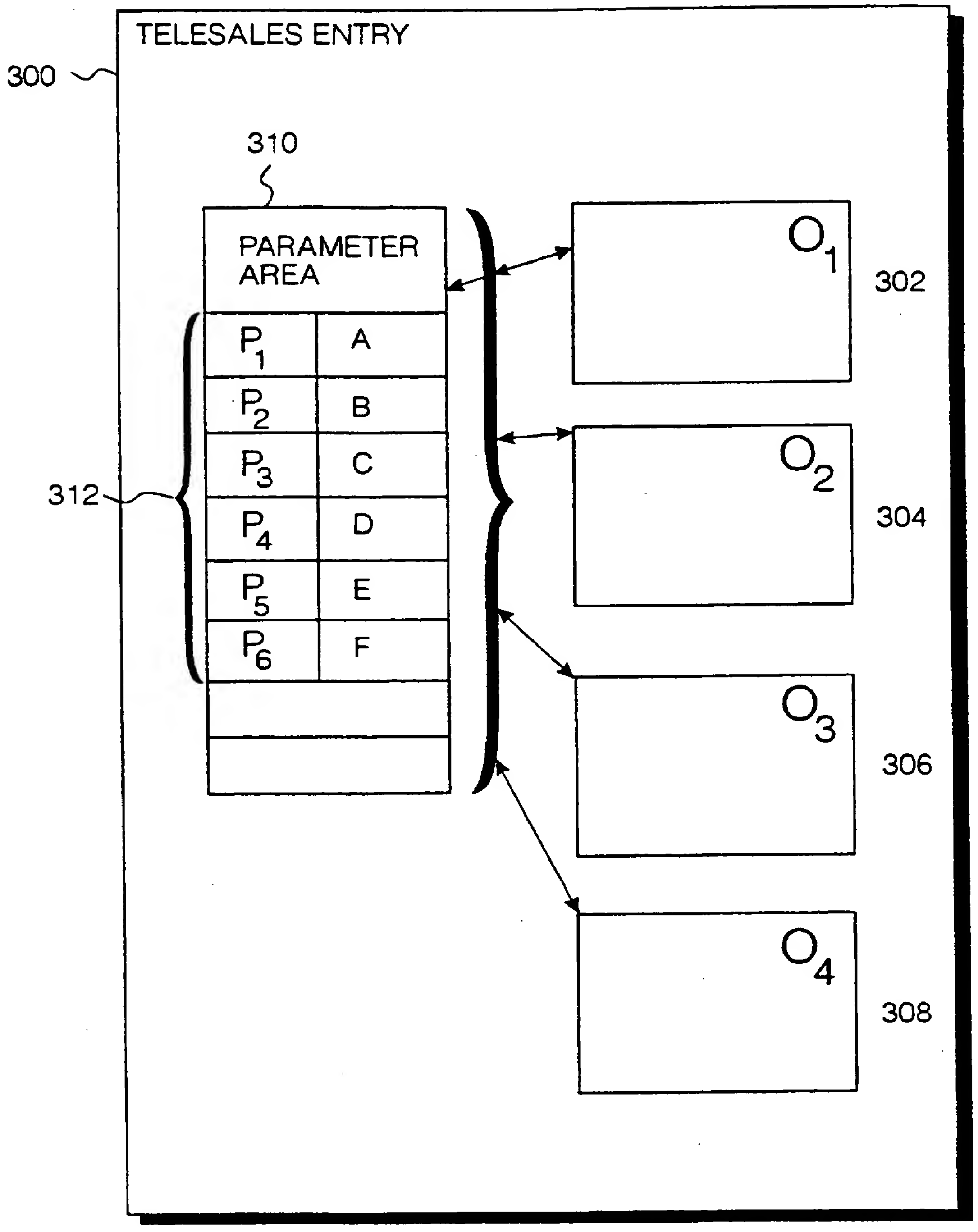

TELESALES ENTRY
300
310
PARAMETER AREA
P_1 A
P_2 B
P_3 C
P_4 D
P_5 E
P_6 F
312
O_1
302
O_2
304
O_3
306
O_4
308

Fig. 3

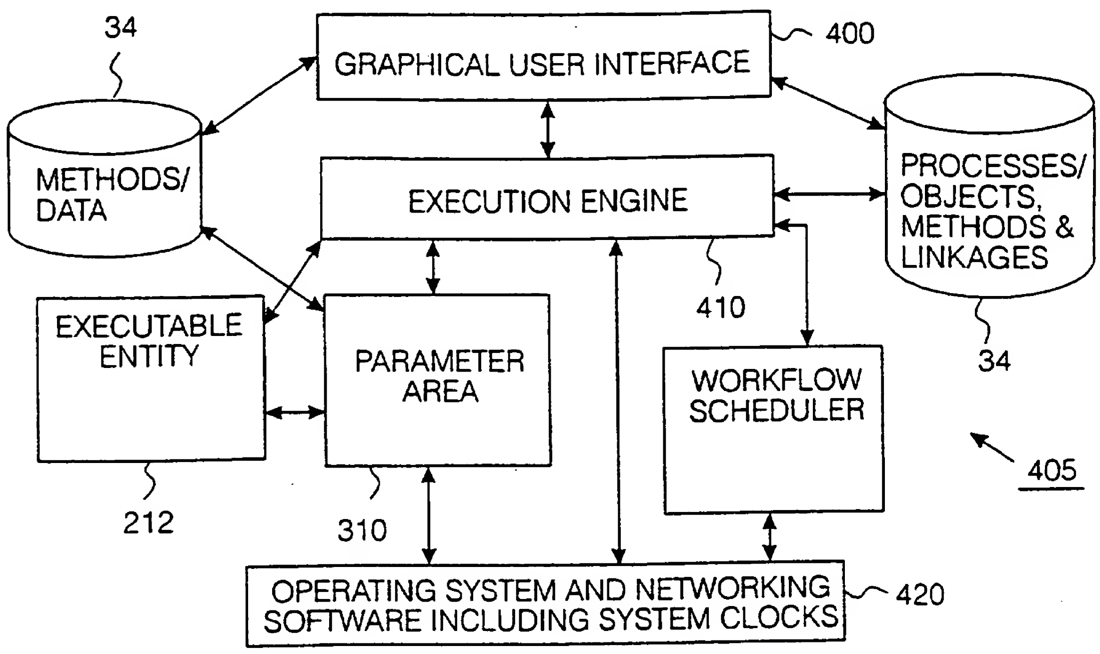

34
GRAPHICAL USER INTERFACE
400
METHODS/ DATA
EXECUTION ENGINE
PROCESSES/ OBJECTS, METHODS & LINKAGES
410
34
EXECUTABLE ENTITY
PARAMETER AREA
WORKFLOW SCHEDULER
405
212
310
OPERATING SYSTEM AND NETWORKING SOFTWARE INCLUDING SYSTEM CLOCKS
420

Fig. 4

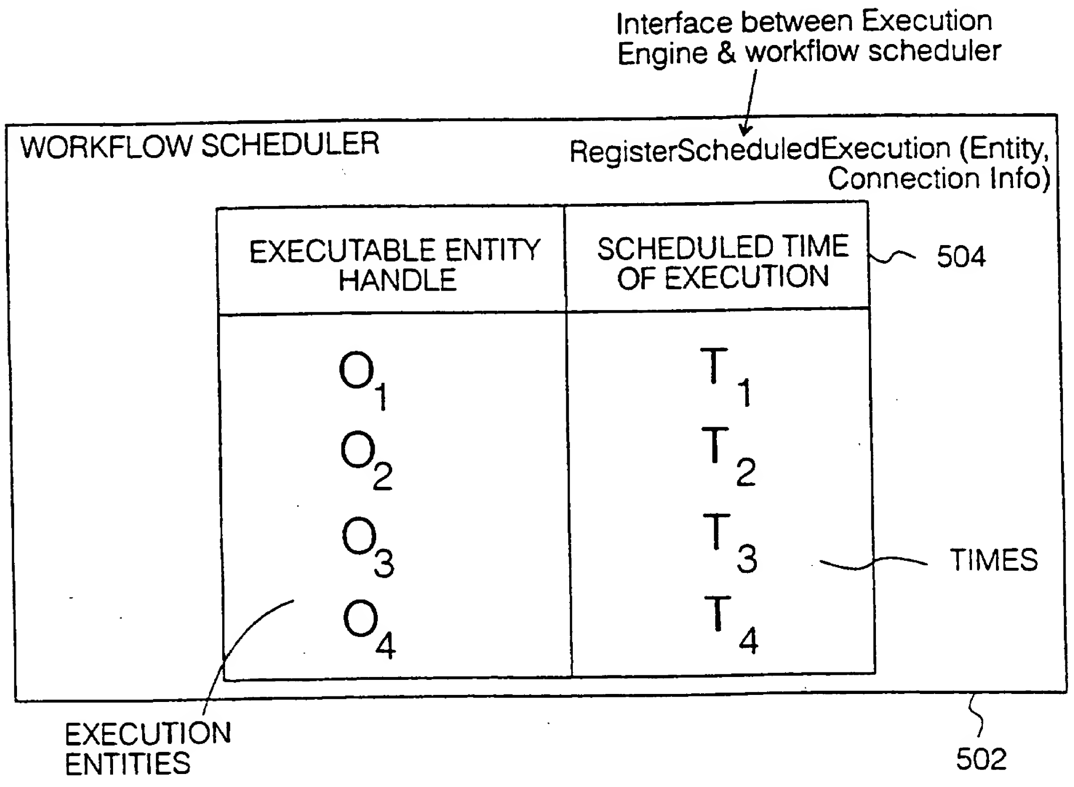

Interface between Execution Engine & workflow scheduler
WORKFLOW SCHEDULER
RegisterScheduledExecution (Entity, Connection Info)
EXECUTABLE ENTITY HANDLE
SCHEDULED TIME OF EXECUTION
504
O1 T1
O2 T2
O3 T3
O4 T4
TIMES
EXECUTION ENTITIES
502

Fig. 5

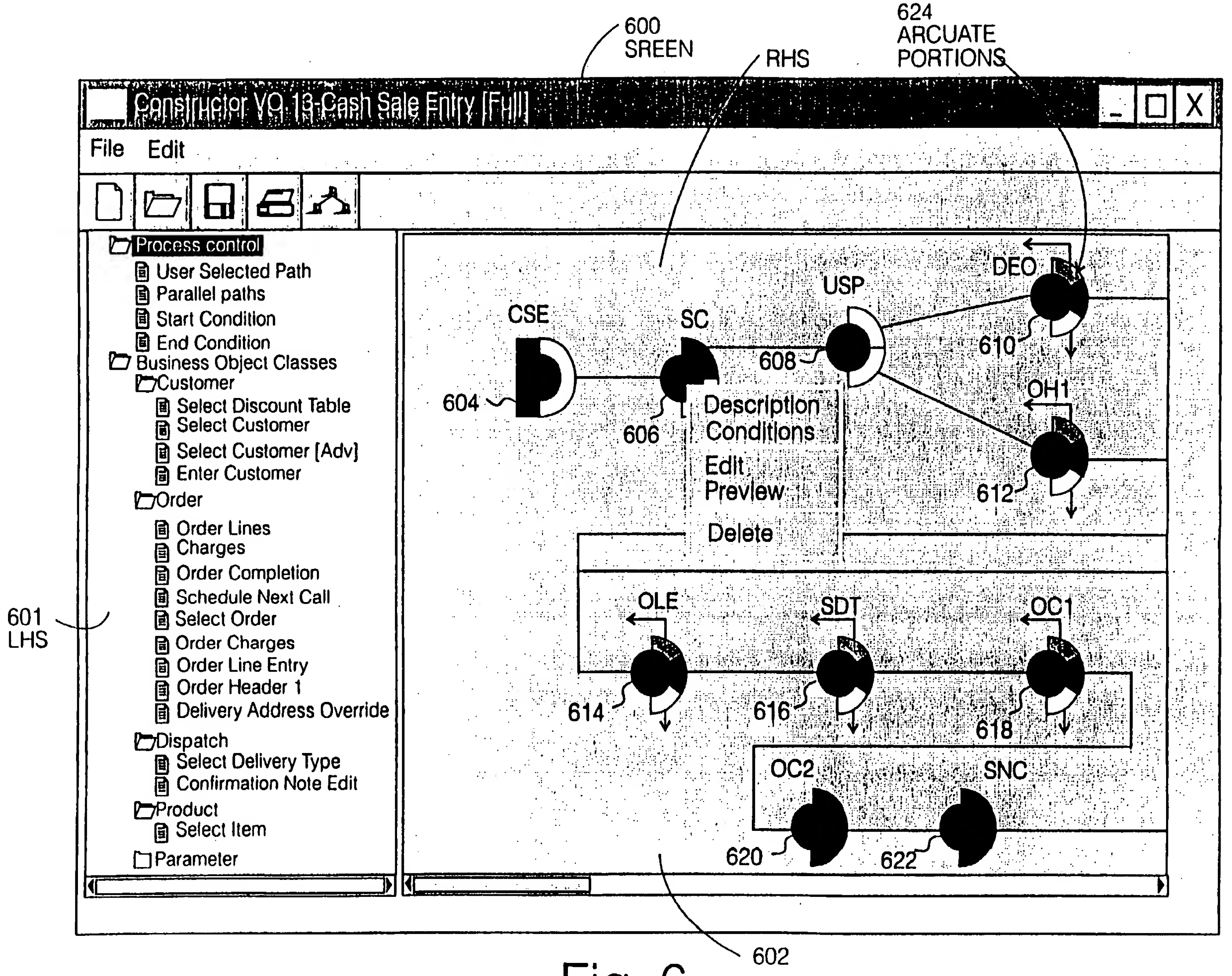
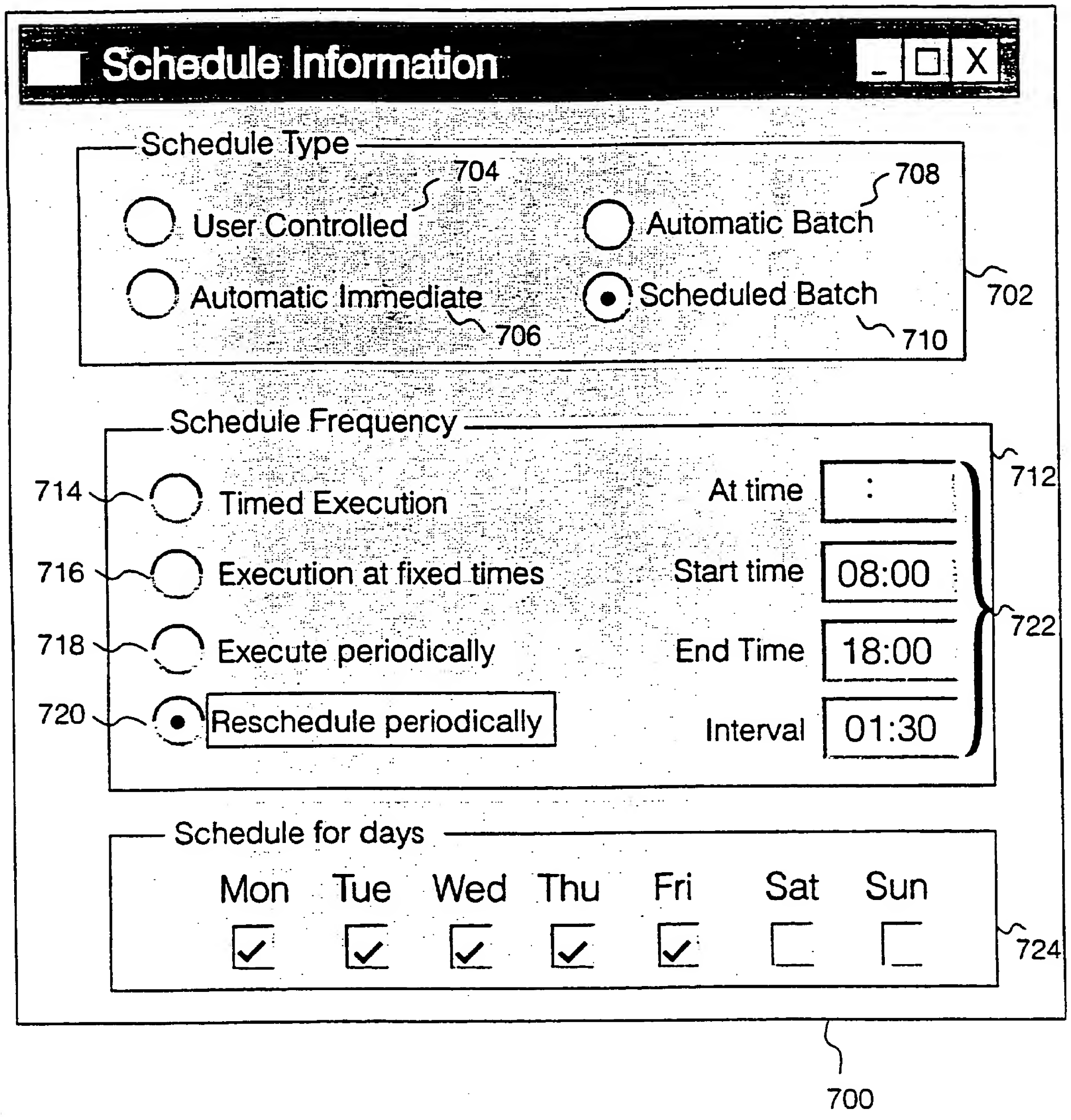

600
SREEN
RHS
624
ARCUATE
PORTIONS
Constructor VO 13-Cash Sale Entry [Full]
File
Edit
Process control
User Selected Path
Parallel paths
Start Condition
End Condition
Business Object Classes
Customer
Select Discount Table
Select Customer
Select Customer [Adv]
Enter Customer
Order
Order Lines
Charges
Order Completion
Schedule Next Call
Select Order
Order Charges
Order Line Entry
Order Header 1
Delivery Address Override
Dispatch
Select Delivery Type
Confirmation Note Edit
Product
Select Item
Parameter
601
LHS
CSE
604
SC
606
USP
608
DEO
610
OH1
612
Description
Conditions
Edit
Preview
Delete
OLE
614
SDT
616
OC1
618
OC2
620
SNC
622
602

Fig. 6



Schedule Type

( ) User Controlled [✓ Scheduled Batch] ( ) Automatic Batch ( ) Automatic Immediate

Schedule Frequency

( ) Timed Execution ( ) Execution at fixed times ( ) Execute periodically [✓ Reschedule periodically]

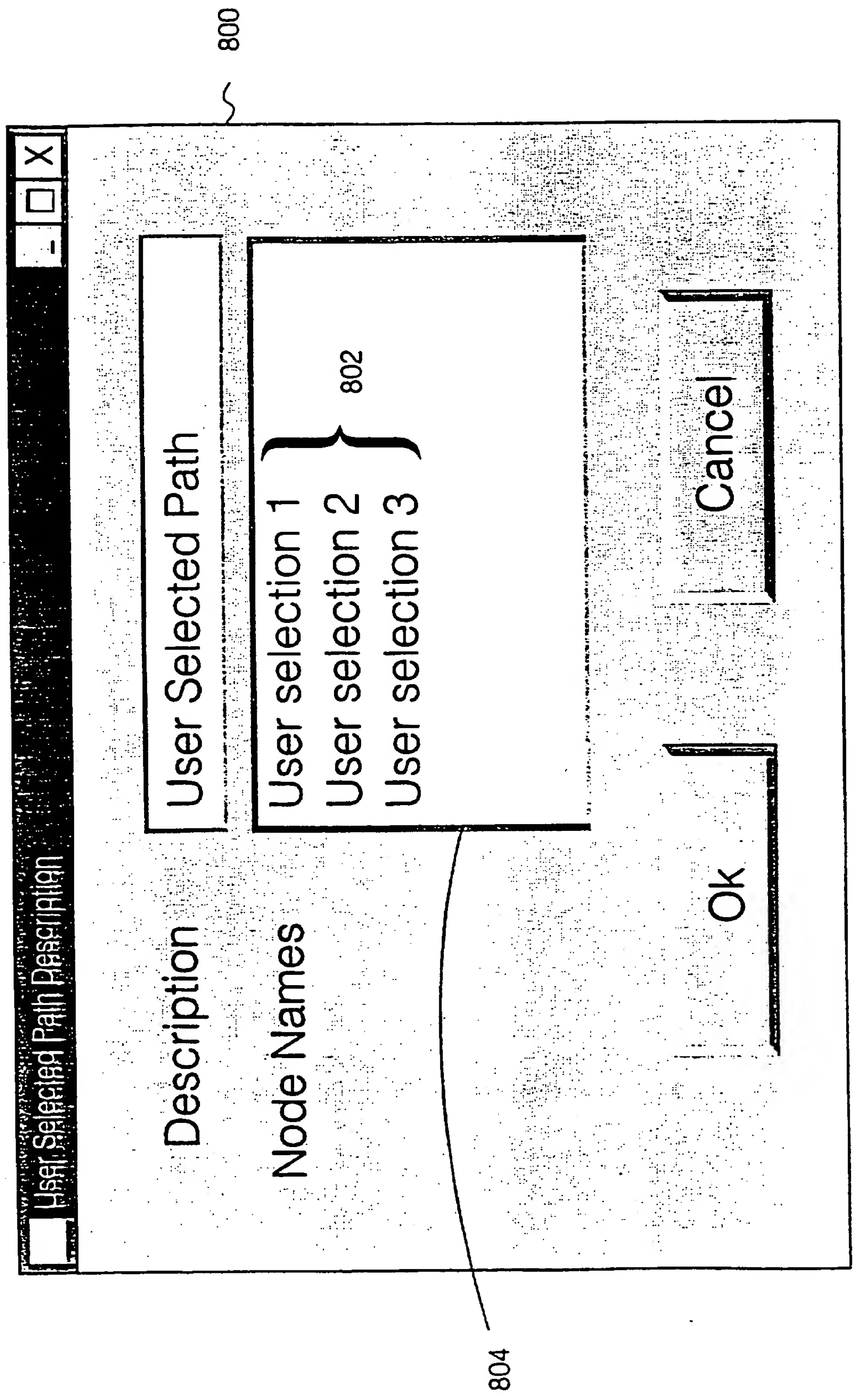
At time	:
Start time	08:00
End Time	18:00
Interval	01:30

Schedule for days

Mon	Tue	Wed	Thu	Fri	Sat	Sun
✓	✓	✓	✓	✓		

Fig. 7

8/8



Description: User Selected Path

Node Names:
User selection 1
User selection 2
User selection 3

Fig. 8

2320111

DATA PROCESSING SYSTEM AND METHOD

The present invention relates to a data processing system and method.

The design of computer software implementing business systems is invariably a very complex and difficult task which requires the skills of a complex team including, for example, skills of a systems analyst to produce or to conduct a detailed analysis of the business processes to be implemented or the current implementation of those business processes. That analysis is then, typically, passed onto a team of programmers who will produce the code or software necessary to implement the business system or processes.

It will be appreciated that encoding of the software in a computer language, such as visual basic or an object-oriented language such as, for example, C++, requires a great deal of technical skill and knowledge. Such tasks are undertaken by programmers skilled in the relevant language.

While a computer programmer may have a significant degree of technical skill and knowledge insofar as concerns producing software or writing software, it does not necessarily follow that that computer programmer also has a significant and global understanding of the business processes to be implemented by that software. It is for this reason, amongst other reasons, that the design and production of computer software can take a longer period of time than initially anticipated.

Furthermore, the lack of understanding of the business processes to be implemented on the part of either systems analyst or the computer programmer can also lead to incorrect coding or to software which does not meet the requirements of the business.

For very large complex business processes, a team of computer programmers is invariably employed to implement the design requirement specified by the systems analyst. Once again, it would require a significant degree of co-ordination between the programmers or control by the project managers to ensure that the various modules or objects written by each computer programmer interact and co-operate in the intended manner with various objects or modules written by other members of the programming team. The net result is that the computer software so developed or so written may not meet the business requirements of the businessman.

As stated above, the actual encoding of computer software invariably requires detailed knowledge of a particular programming language and the associated compilers and linkers thereof. Clearly, every businessman, irrespective of their degree of competency at programming, cannot concern himself with the detail of encoding computer software; it is not an effective utilisation of his time. It does not necessarily follow that the businessman designing a business process will be well versed in any particular computer language. Therefore, it is necessary for the businessman to retain the services of a systems analyst and team of computer programmers to computerise his business processes.

Furthermore, many software applications are implemented by way of a single large program which addresses or attempts to address all of the requirements of the businessman. Such large or monolithic software entities have the disadvantage that they are difficult to modify in the event that the business processes implemented are subject to change. Any such change would require accompanying changes to be made to the software. The changes can be extensive and may represent a significant drain on resources, both financial and in terms of manpower.

The above problem stems from the inflexibility of monolithic programs. An attempt has been made to address the above by providing a single large program which contains software switches, or flags, to determine which parts of the program can be executed. However, it is not often that software is designed with such flexibility in mind. Furthermore, the order of execution of the constituent parts of the software cannot be fundamentally modified.

Object-Oriented programming languages, such as C++ and the like, represent a further attempt to increase or improve the flexibility and re-useability of software or software components. The operation of software objects is well understood by those skilled in the art. Each object can be arranged to perform a particular task. Objects interact with one another using a message based paradigm, that is to say, messages are passed between objects to invoke the methods of the object receiving the message. This collocation and interaction between objects achieves the ultimate end of the software.

Passing such messages or parameters between objects requires a calling or transmitting object to know the parameter list structure of the called or receiving object. Clearly, if a change to the order of execution of the objects was required, the parameter definitions of each object would have to be changed accordingly. Such a change would, again, represent a significant re-development effort.

It is an object of the present invention to mitigate the problems associated with the prior art.

Accordingly, a first aspect of the present invention provides a method for processing data in or configuring a data processing system, said data processing system comprising memory for storing a plurality of executable entities, means for establishing an order of execution of selectable ones of the executable entities, and means for executing the selected executable entities according to said order, without modification of said selected executable entities, the method comprising the steps of storing, in said memory, a plurality of executable entities, establishing an order of execution of selectable ones of said executable entities, and executing the selectable ones of said executable entities according to said order without modification of said executable entities.

The data processing system and method advantageously allow the production of scalable and configurable business software without the need for the businessman to have any knowledge computer programming language.

A preferred embodiment of the present invention provides a method for configuring a data processing system, said data processing system comprising a plurality of executable entities, a memory for storing the plurality of executable entities, a display means for producing a graphical representation of the plurality of executable entities, means for executing said executable entities, input means for selecting for execution selectable ones of said plurality of executable entities, the method comprising the steps of displaying said plurality of executable entities, selecting two of the executable entities for execution, establishing an order of execution for the two executable entities, and executing the two executable entities according to said order without modification of said two executable entities.

A still further feature of the present invention provides a method further comprising the steps of establishing, within said memory, a data structure, commonly accessible by said selected executable entities, which is arranged to allow said selected executable entities to store therein and retrieve therefrom any data object and/or to allow indirect exchange of data between said selected executable entities.

Yet another feature of the present invention provides a method wherein said data processing system further comprises an execution engine which instigates the execution of plurality of executable entities, and wherein the step of executing the executable entities comprises the steps of providing each of said plurality of executable entities with a common interface for instigating execution of the executable entities, and instigating, by said execution engine, execution of said two executable entities utilising said common interface.

A second aspect of the present invention provides a data processing system comprising memory for storing a plurality of executable entities, means for establishing an order of execution of selectable ones of the executable entities, and means for executing the selected executable entities according to said order without modification of said selected executable entities.

A feature of the second aspect of the present invention provides a data processing system further comprising means for establishing, within said memory, a data structure, commonly accessible by said selected executable entities, arranged to allow said selected executable entities to store therein and retrieve therefrom any data object and/or to allow indirect exchange of data between said selected executable entities.

Preferably, the present invention provides a data processing system for configuring control means of said data processing system, said control means comprising a plurality of executable entities, a memory for storing the plurality of executable entities, a display means for producing a graphical representation of the plurality of executable entities, means for executing said executable entities, input means for selecting for execution selectable ones of said plurality of executable entities, the system comprising means for displaying on said display means a graphical representation of said plurality of executable entities, means for selecting at least two of the executable entities for execution, means for establishing an order of execution for said at least two executable entities, and means for executing said at least two executable entities according to said order without modification of said at least two executable entities.

The present invention further provides a data processing system further comprising an execution engine which instigates the execution of said plurality of executable entities, and wherein each of said plurality of executable entities comprises a respective interface having a common format and wherein said means for executing the selected executable entities utilises the respective interface of an executable entity.

It will be appreciated by those skilled in the art that the distribution of software can be effected by either pre-loading the harddrive of a computer with that software or by using other magnetic storage media, such as diskettes. Still further, CD-ROMS can also be used as the means of distribution for software.

Accordingly, a third aspect of the present invention provides a computer program product for processing data in or configuring a data processing system, said data processing system comprising memory for storing a plurality of executable entities, means for establishing an order of execution of selectable ones of the executable entities, and means for executing the selected executable entities according to said order, without modification of said selected executable entities, the computer program product comprising computer readable program code means for storing a plurality of executable entities, computer readable program code means for establishing an order of execution of selectable ones of said executable entities, and computer readable program code means for executing the selectable ones of said executable entities according to said order without modification of said executable entities.

Preferably, the further aspect of the present invention provides a computer program product further comprising computer readable program code means for establishing, within said memory, a data structure, commonly accessible by said selected executable entities, which is arranged to allow said selected executable entities to store therein and retrieve therefrom any data object and/or to allow indirect exchange data between said selected executable entities.

A feature of the further aspect of the present invention provides a computer program product, wherein said data processing system further comprises an execution engine which instigates the execution of said plurality of executable entities, and wherein said computer readable program code means for executing said at least executable entities comprises computer readable program code means for providing said plurality of executable entities with a respective common interface for instigating execution of the executable entities, and computer readable program code means for instigating execution of said executable entities utilising said common interface.

Further features of the present invention are defined in the claims set out hereafter.

Embodiments of the present invention will now be described by way of example only, with reference to the accompanying drawings in which:

figure 1 shows a computer system suitable for implementing embodiment of the present invention,

figures 2a to 2c illustrates schematically the structure and operation of computer software according to the prior art;

figure 3 illustrates schematically the operation of an embodiment of the present invention by way of a telesales business process;

figure 4 schematically shows the general structure of an executable entity according to an embodiment;

figure 5 shows schematically the functional components of the present invention together with the operating system and networking software;

figure 6 shows a graphical display of a business process according to an embodiment;

figure 7 shows a dialogue box displaying scheduling information; and

figure 8 shows a dialogue box displaying user path selection options.

Referring to figure 1, there is depicted a block diagram of, for example, a computer system 10, such as are commonly known within the art. The computer system includes a 32 bit processor 14 and a system bus 12. Connected to the system bus 12 is a central processing unit (CPU) 14, for example an Intel 80486 or a Pentium Microprocessor. CPU 14 passes data to and receives data from other devices attached to the system bus via the system bus. Traffic on the bus is controlled by a bus controller 16 and an interrupt controller 18 handles interrupts generated by the remaining devices and notifies the CPU 14 accordingly.

The read only memory (ROM) 20 is a non-volatile memory storing power-on test processes and basic input-output systems (BIOS or ROM-BIOS). The system memory 22 is a random access memory into which an operating system, preferably a 32 bit operating multi-tasking operating system such as IBM OS2/WARP or Windows NT which support concurrent processes or multiple concurrent threats, is loaded for execution of computer software applications. Additional hardware components of computer 10 attach to system bus 12 include a memory controller 24 and a system configuration store 26, provided by random access memory (RAM).

Auxiliary data storage is provided by peripheral controllers and associated storage devices including, a floppy disk or diskette controller 28 and a drive 30, a hard drive controller 32 and a hard drive device 34, and a compact disk (CD) ROM controller 36 with CD ROM drive 38. An SVGA controller 40 includes a RAM buffer 44 in which the current frame for subsequent display on a display device is stored. In some computers, the buffer 44 may be loaded directly from system memory 22 or from an auxiliary storage controller.

Direct memory access (DMA) controller 46 handles data transfers between auxiliary storage devices or other input output devices, and system memory 22 without interaction by CPU 14. Keyboard controller 48 provides an interface to a keyboard 50, for user entries, and may be used to provide a max interface. Parallel controller 52 is a device controller for an input or output device (eg a printer connected to computer 10 by a parallel cable). Referring to the figure 5 shown there is schematically shown the functional components of the present invention together with the operating system and networking software.

Referring to figure 2a, there is shown schematically the structure and an operation of conventional computer software. A business process 200, such as tele-sales process, comprises a plurality of executable entities 202 to 208 each of which performs a particular function of the tele-sales business process 200. For example, the first executable entity 202 allows the user of the software to select or input customer information. The second executable entity 204 allows the user to input order lines or order information relating to goods or services required by the customer. An order line may comprise an item code, which identifies an item to be purchased, an item description and a quantity of the items to be despatched or ordered. The third executable entity 206 provides for the entry of charging information relating to the various goods ordered and the final executable entity 208 completes the order.

Each of the executable entities, in order to be able to interact with neighbouring executable entities passes parameters P_1 to P_6 between one another. Therefore, a parent executable entity, for example executable entity 202, must known which parameters are required by the child executable entity 204 for the correct execution of the latter.

It will be appreciated that if modification of the order of processing of the tele-sales process is required, significant changes to the software, in particular the parameter passing or interfaces between the executable entities, will be required. For example, if a business process was implemented comprising an executable entity which requests information relating to a delivery address for ordered goods and an executable entity which requests information relating to delivery charges appropriate to that delivery, executed in that order, and it was necessary or desirable to change that order of processing such that the delivery charges are determined before the input of the delivery address, it would be necessary to ensure that an executable entity, such as, Select Delivery Charges passes parameters to the executable entity Select Delivery Address. For example, it is likely that prior systems would pass an indication of the delivery address to a function which requests or determines the delivery charge.

It is also likely that changes to the methods of the executable entities will also be required.

Re-sequencing of software or re-structuring of software typically within an object-oriented environment presents particular difficulties. Referring to figure 2b, there is shown schematically an illustration of the property of inheritance and the derivation of subclasses 210 and 212 from a parent class 214. It is necessary for the interfaces I_{12} and I_{23} between the various objects to be defined so as to allow data to be passed across the interfaces I_{12} and I_{23} to the various levels of the object. It would not be possible to re-sequence to objects as per figure 2c in which the parent object 216 is object 2 and the derived objects 218 and 220 are objects 3 and 1 respectively without modification of the class definitions and the interfaces I_{32} and I_{13} between those objects, that is to say, without modification and recompilation of the underlying code.

Referring to figure 3, there is shown schematically a tele-sales business process 300 according to the present invention comprising a plurality of executable entities 302 to 308, and a parameter area or object 310 for storing data created and/or manipulated by the executable entities 302 to 308. The order of execution of the executable entities 302 to 308 constituting the tele-sales business process 300 can be readily changed without requiring concomitant changes to the executable entities 302 to 308.

The executable entities are self-contained and comprise methods which are ready to run without further compilation or, alternatively, immediately ready to interpret, if an interpretive language has been utilised. An executable entity can be, but is not limited to, an object, a function, another self-contained program or a DLL task.

An object or executable entity comprises a plurality of elements. Each element has a unique identification code associated therewith.

Table 2 below illustrates a pseudo-code for an embodiment of an executable entity.

```
DLLName::ClassName      ;Unique Identifier
        Execute         ;Execute Method for invoking Entity
        Preview         ;Preview Method for testing
        Other Methods   ;Methods which are invoked from
                        Execute defining the Entities
                        behaviour
```

TABLE 2

The DLLName represents the Dynamic Link Library containing the class definition for an object or executable entity. "Execute" represents an interface with an executable entity. The interface has a common format for all executable entities thereby presenting a consistent manner of invoking or executing an executable entity. "Preview" allows execution of an entity in isolation using default test data and independently of any other entity. The "Other Methods" are invoked by Execute and further define the behaviour of the object.

It will be appreciated that in a distributed system, the executable entities may not all reside on a single server or machine. Therefore, an embodiment can be realised in which the unique identifier of the class definition for the executable entities further comprises an indication of the server upon which the DLL containing the class definition is resident.

Each object, upon, execution can return at least one of a number of terminating conditions. A terminating condition represents the result or outcome of the execution of an associated object and determines which of a plurality of execution paths from the executable entity are followed. This governs the flow of control throughout a business process of collation of executable entities.

For example, a dialogue box may have been presented to a user which contained a number of options together with an invitation to select one of the options. The terminating condition would be determined by the selected option and therefore determine the executable entity which is executed next.

Further, each executable entity has an associated array of connections. There is one connection or link per terminating condition.

Most of the executable entities can be executed independently of any of the other executable entities.

Each executable entity accesses data required for correct operation via the parameter area 310. The parameter area is an extendible data structure to which data or object of various sizes and types can be added, updated or deleted. The parameter area 310 is dynamically created using a Visual Basic command called New Collection. The operation of this command is well known within the art and described in further detail in the MicroSoft Visual Basic Version 4 online reference manual the entire content of which is by reference incorporated herein.

The parameter area 310 comprises a plurality of entries 312. Each entry comprises an identification field by which a parameter P_1 to P_6 can be identified and therefore a corresponding value A to F cab be accessed.

An executable entity accesses a value associated with a parameter by calling the parameter area using the Visual Basic function Get Parameter using the parameter identifier as a key. The function Get Parameter returns the value associated with the parameter or key.

For example, if an executable entity O_3 required access to the value D associated with a parameter P_4, the call to the parameter area 310 would specify "P_4" as a parameter. The parameter area 310 would return the associated value D of a parameter P_4 to the calling executable entity O_3.

The parameter area is commonly accessible by the executable entities 302 to 308 of a business process, such as that defined by the tele-sales process 300. It can be seen that the executable entities do not directly exchange data as per conventional programming methods or according to conventional object-oriented technologies. The executable entities according to the present invention exchange data indirectly via the commonly accessible parameter area. Therefore, the requirement for parameter passing between executable entities is removed.

It follows as a consequence of the indirect data exchange between executable entities that a change to the order or sequence of execution of the executable entities does not necessitate corresponding changes to the executable entities. This is in contrast to the direct interfaces and parameter passing techniques utilised in the prior art.

Preferably, an embodiment, an execution engine is provided which instigates execution of the plurality of executable entities according to the execution order specified by the links therebetween.

The execution engine establishes an identifier or handle for each invocation or instance of an executable entity. The handle or identifier is used to access and instigate execution of the methods of the executable entities.

The executable entities are implemented as Visual Basic Objects. The methods of the objects are invoked, assuming an instance of an object is called "ExecuteElement" and the method of the object is called "Execute", by the execution engine as follows: "ExecuteElement.Execute".

A business process is implemented using the Visual Basic Collection Object facility. A Collection Object allows the creation of a complex entity from simpler executable entities. The operation of the Visual Basic Collection Object is well understood by those skilled in the art and is further described in the Visual Basic Version 4 online reference manual. The parameter area can also be implemented using a Collection Object. Alternatively, the parameter area can be implemented using a simple two dimensional array together with means for searching the array to retrieve data therefrom.

The method "Execute" within each executable entity instigates execution of the object and allows an executable entity to present a common interface structure to the execution engine.

Due to the common interface structure presented by each object of a business process, the execution engine can instigate execution of all or any objects independently of the function performed by that object.

Furthermore, as the parameter area stores the data required by an executable entity or the methods thereof, the execution engine does not need to pass such data or parameters to the executable entity.

Furthermore, as parameter passing between objects has been obviated or obliterated by the use of external or separate data or parameter area, that is the Collection Object of Visual Basic, it is not necessary to pass directly parameters between the executable entities.

An executable entity may be connected to or logically followed by a plurality of other executable entities. The next entity for execution is determined according to the result of the execution of the current executable entity. Each executable entity produces, upon termination, a termination condition which provides an indication of which executable entities should be executed next, that is to say, the termination condition identifies a link leading to the next executable entity to be executed. The termination condition can be a simple number or, more commonly, an element of an enumerated type.

Each link has an identifier associated therewith and each executable entity contains a data structure, such as a linked list or array, comprising the identifiers of links leading therefrom. The execution engine controls the flow of processing according to the returned or produced termination conditions. In some instances, the executable entity may present a dialogue box to the user of the data processing system which requests the user to select at least one of a plurality of preferred courses of action. The input then determines the link followed and hence the next executable entity to be executed.

It will be appreciated that a business process comprises a plurality of executable entities, all interlinked to form a complex graph or network of executable entities. Execution of the business process should, preferably, commence with the first, or start, entity in the graph. The start entity has associated therewith an indication of the type of that entity. Entity types include "START" and "END" types which indicate respectively that an associated entity represents the first or starting entity of a business process or the end or terminal entity of a business process.

Once the execution engine has located the start node or entity, execution of the start entity, and hence the overall business process, can be commenced.

Similarly, the last executable entity in one of the possible execution paths through the graph of connected entities carries an identifier designating that executable entity as an End or Terminal Entity. It will be appreciated that the graph of executable entities will have at least one terminal node and in many instances may have a plurality of Terminal or End Entities.

The execution engine identifies or accesses executable entities using a handle of an instance of an executable entity and instigates the execution of the methods of that entity as described above using that handle. When the execution of that entity terminates, the termination condition is determined, the link associated with that termination condition is determined and hence the next executable entity for execution is also determined.

Processing or traversal of the graph of executable entities continues until an "END" type entity has been completed whereupon processing terminates.

Optionally, each executable entity comprises a further type, namely a connection mode type, which further governs the behaviour of that entity. The connection mode type can take one of two values, "IMMEDIATE" or "SCHEDULED". The purpose of these types is to determine, upon termination of execution of one entity, whether or not execution of the following entity should commence immediately or only after a predeterminable time period or at a predeterminable point in time. If the connection mode type is "IMMEDIATE" then processing of the next executable entity commences immediately. However, if the connection mode type is "SCHEDULED" then the next executable entity is added to a table of other executable entities scheduled for execution after a predeterminable period of time has elapsed or after at least one of a predeterminable number of conditions have been satisfied.

Referring to figure 4 there is shown schematically the structure of the elements of a data processing system according to an embodiment. The system 405 provides a graphical user interface (GUI) 400 capable of displaying the selectable plurality of self-contained, precompiled executable entities. The executable entity 212 currently being executed is also shown. Each entity represents part of a business process or a self-contained business process. The plurality of executable entities are stored on a suitable storage medium, such as a hard disk drive device 34 and 32. The graphical user interface 400 interfaces with the execution engine 410. The execution engine functions as described above. Similarly, the executable entity functions substantially as described above.

The GUI 400 allows a businessman or a user of the data processing system to select various executable entities stored on the hard drive 34 in order to construct therefrom a more complex business process using a drag and drop regime as is well know within the art. Therefore, a number of basic or simple executable business processes can be arranged or utilised to produce a more complex process. These more complex processes can, in turn, be used to larger software business systems.

The user of the data processing system establishes links between the various executable entities using a mouse and one of the selection keys thereof. A link is established between two entities, a parent entity and a child entity, by clicking on the parent entity and subsequently clicking on the child entity.

The links define the order of execution of those executable entities and hence the overall operation of software which is intended by a businessman to process data according to his business processes.

During the design of computer software for the business process, the processes and methods and linkages therebetween are stored in a software bin and graphically displayable.

When it is desired to execute a constructed business process, the executable entity and associated linkage information is interpreted by the execution engine 210. During execution of the entities using the associated linkages, the execution engine 210 creates the parameter area 310, implemented using either a simple two-dimensional array or a Collection Object as provided by

Visual Basic. The parameter area 310 is used to store data accessed and manipulated by the executable entities.

Finally, the operating system and network working software 420 co-ordinates the utilisation of the computer system resources by the various functional entities, such as the graphical user interface 400 and execution engine 410, and also the management of the memory for storing the parameter area.

Optionally, a link entity can be implemented as a more specific form of a general executable entity. In the optional realisation, a link entity contains methods and/or data which defines the behaviour of or interaction between that link any executable entities attached thereto. The data stored within a link is utilised by the execution engine to determine the other executable entity or entities to which it is connected. Each link has an link identifier, a parent executable entity identifier, a child executable entity identifier, and timing information and methods which define or govern the behaviour of the link and the interaction thereof with parent and child executable entity executable entities.

Referring to figure 5 there is shown schematically a workflow scheduler 502 containing a table 504 of executable entities O_1, O_3, O_2, and O_4 scheduled for execution at respective specified times T_1, T_2, T_3 and T_4.

The workflow scheduler 502 manages the scheduling of the execution of executable entities. The stored information comprises the handle of an executable entity together with an indication of the scheduled execution time or the conditions which have to be satisfied before execution will be commenced. The information required for scheduling an executable entity is provided by the execution engine 410 via an interface RegisterScheduledExecution(Entity,ConnectionInfo). This interface represents the common interface presented by each executable entity and which is utilised by the execution engine 410.

Alternatively, execution of an executable entity may not be commenced until the status of various parameters in the table have assumed a predeterminable values.

If the data processing system comprises an operating system which supports multiple threads, a timer task can be instigated for each entity which is scheduled to commence processing at or after a specified or predeterminable period of time. This task can then interrupt the execution engine 410, supply the entity handle to the execution engine 410 which can then call the Execute method of that executable entity. Similarly, if a predeterminable condition has to be satisfied or a predeterminable number of variables/flags have to be set before execution can commence, then respective processes or tasks can be created using respective threads which check the conditions or status of the flags and which interrupt the execution engine accordingly.

Alternatively, the workflow scheduler 502 can periodically scan through the table 504 of scheduled entities to perform the above checks and instigate execution of the entities according to whether or not the above conditions have been satisfied.

In a still further embodiment, the periodic execution of an executable entity can be arranged thereby enabling the entity to be executed once every 5 minutes for example. A publicly accessible variable "IncTime" of ConnectionInfo is used to indicate whether or not periodic execution is required. If the variable "IncTime" contains a NULL value, then periodic execution is not required. However, if "IncTime" contains some other value, periodic execution will be established according to a time period governed by that other value. Once execution of such an entity has been completed, the workflow scheduler automatically re-schedules future execution of the entity according to the indicated time period.

The time of execution may be specified in terms of a relative offset to the current time or in terms of absolute time. The format of the time may be in terms of Year, Month, Day, Hour and Minutes.

The pseudo-code shown below in Table 1 provides an indication of the flow of control of an execution engine according to an embodiment. The line numbering is for ease of reference only.

```
1    DataObject = New Collection
2    ElementInfo = FindStartElement(Entity)
3    While ElementInfo <> ElementOfTypeEnd
4        ExecuteElement =
         CreatObject(ElementInfo.Classname)
5        ExecuteElement.Execute; write to
         DataObject.Terminate how execution terminated
6        ConnectionInfo =
         FindConnectionMatchingTerminateCondition;
7        If ConnectionInfo.Mode = ImmediateExecution
             ElementInfo = ConnectionInfo.ElementInfo
         else
             RegisterScheduledExecution(Entity,Connec
             tionInfo)
             exit While
8        endif
9    end while
```

TABLE 1

"DataObject" is an object which is utilised to store temporarily information associated with an executable entity such as the termination condition or state of the executable entity. This termination condition is stored in a publicly accessible variable called "Terminate".

"ElementInfo = FindStartElement(Entity)" determines the handle of the executable entity which is the START entity in the graph of entities. The information stored by ElementInfo optionally comprises a plurality of public variables. The variables including an indication of whether or not the executable entity is of a type "END" and "ClassName" which identifies the class definition or structure of the object. Other public variables are "NodeConnections As Collection", which is a collection or array of the output connections or links from that entity to other executable entities, "Parms as Objects", which identifies the data parameters which can be shared between the methods of an executable entity, "ConditionString", which identifies a list of conditions which should be satisfied before execution of the methods of the entity can commence, and "DLLFUNCTION" which stores the name of an external program which can be accessed and executed from within the executable entity.

Optionally, other public variables comprises screen co-ordinates which determine the position of a graphical representation of the executable entity. Further information regarding the graphical user interface is set out below.

"ConnectionInfo" comprises a public variable named "Mode" which is used to indicate whether the executable entity should be executed immediately or whether the executable entity should be scheduled for execution at a particular time. It can be seen that line 7 determines whether or not the execution of the entity needs to be commenced immediately or scheduled for a particular time.

If scheduling is required, a call is made to a method of the workFlow scheduler 502 called "RegisterScheduledExecution" passing parameters identifying the executable entity and the associated connection information.

The execution of the while-loop continues until either an executable entity is scheduled for execution or until an END entity has been executed.

Referring to figure , there is shown schematically the layout of the screen presented using the GUI 400. The GUI 400 operates in accordance with defined MicroSoft Windows Standard as is well known within the art.

The screen 600 is divided into two parts. The left-hand side 601 depicts in the form of a tree the list of available business object classes or processes, activities and methods (executable entities) constituting the business processes. The right hand side 602 of the screen 600 depicts the executable entities constituting the business process.

It can be seen that the business process comprises ten executable entities selected from the tree structure of the left hand side of the screen.

The executable entity "Cash Sale Entry" 604 is the start node for the graph of execution entities. The end node is not shown in this example. The executable entities are linked via links substantially as described above which define the flow of execution of the executable entities.

Once the execution engine has located the start node 604 execution of the process can commence. The entity entitled "Select Customer" 606 is executed so as to select one of a plurality of possible waiting customers. The customers may be held in a telephone queue or may have details previously stored in file capable of being displayed upon invocation of this entity 606 or business process.

Having selected the customer whose order is to be processed, the execution passes to the next executable entity 608. In the instant case, the next executable entity 608 requires input from the user of the computer system. The "User Selected Path" entity 608 allows the user of the data processing system to select at least one of a possible plurality of execution paths from the entity. In the instant case, two possible execution paths or links are provided. The first link leads to an entity called Delivery Address Override 610 which allows a default address usually associated with the selected customer to be overridden with a different deliver address. The second link leads to an entity Order Header 1 612 which requests from the user typical order header information.

Processing continues to the entity called Order Line Entry 614 which enables the user of the data processing system to input the actual order information taken from the selected customer. Having input the order, execution continues to the next entity, Select Delivery Type 616, which allows the user to select the mode of delivery for the order. The modes of delivery include courier and various types of normal postal services.

The charges for the goods associated with the order are then calculated via the executable entity called Order Charges 618. The Order charges executable entity 618 can have access to external pricing software which governs the calculation of the costs associated with an order according to different and selectable pricing regimes. Similarly, the various pricing routines may also take into account the currency in which the charges are to be furnished.

The Order Completion 620 executable entity then allows the user to confirm that the order is completed. The Schedule Next Call entity 622 allows the user to repeat the above process for another selected user.

A business process is constructed by selecting an executable entity from the left-hand side of the screen and placing in the right hand side of the screen using a drag and drop action with the mouse. The links between the executable entities are established again using the mouse and a rubber-banding technique. A link is established between two entities by selecting one entity using the left mouse button and then moving the cursor to another entity to which the first is to be connected while keeping the mouse button depressed. Upon release of the mouse button a link is displayed between the two executable entities. As described above, the executable entities contain an array for storing identifiers of executable entities linked thereto. Therefore, during the rubberbanding operation, the identifier of the second executable entity is stored within the array.

The graphical depiction of an executable entity comprises a coloured arcuate portion 624 which represents a link or connection to another executable entity. It can be seen from the Delivery Address Override 610 entity that the top most arcuate portion has emanating therefrom an arrow pointing to the left. If the user selects this output connection, the processing associated with the corresponding executable entity is repeated.

The down arrow emanating from the bottom most arcuate portion of the segmented area indicates a cancel selection option which interrupts the execution of the executable entity and restores any data values which were altered during execution.

As described above in relation to the publicly accessible "Conditions" variable of the ElementInfo object, each executable entity has associated therewith an indication of the global data which must exist for before an executable entity can sensibly operate. Using the Condition information for each executable entity within a process it can be determined whether or not the completed or constructed business process has a meaningful flow of execution. The execution engine checks all input conditions of executable entities and determines whether or not these conditions would, during normal execution, have been satisfied by the execution of components prior to it. This allows an assessment of the flow of control of a business process to be evaluated without having to actually run the complete business process, that is without having to execute all entities constituting a business process.

Referring again to figure 600, it is optionally possible to display information relating to the scheduling information of executable entities. For example, "clicking" using the left mouse button on a displayed link produces a dialogue box such as that displayed in figure 7. The dialogue box 700 displays the scheduling information of an executable entity. It can be seen that various types of scheduling are available, for example, immediate execution, batch execution, or periodic execution by time, day or both.

With reference to figure 700, in particular the area "Schedule Type" 702, it can be seen that four types or modes of scheduling are provided, that is to say, "User Controlled" 704, "Automatic Immediate" 706, "Automatic Batch" 708 and Schedule Batch" 710.

When an executable entity is encountered in which scheduling is specified as being "User Controlled" 704 the user of the system is presented with a dialogue box as shown in figure 700 and requested or expected to complete the area of the screen designated "Schedule Frequency" 712. This action sets the scheduling time and governs the time of execution of the respective executable entity.

If an executable entity is scheduled for "Automatic Immediate" that executable entity is executed substantially immediately.

Two types of batch mode execution are provided, namely "Automatic Batch" 708 and "Scheduled Batch" 710. Batch mode operation utilises a thread running as a background task and execution of an executable entity which is designated for batch mode execution is performed using that background thread. The "Automatic Batch" 708 designation indicates that execution of the associated executable entity should commence immediately on the background thread. In contrast, a "Scheduled Batch" 710 designation indicates that execution of the associated executable entity should only be commenced at the appropriate scheduled time on the background thread.

Furthermore, the frequency of scheduling can be selected. The options according to the present embodiment include "Timed Execution" 714, "Execute at fixed Times" 716, "Execute Periodically" 718 and "Reschedule Periodically" 720 together with respective time entry fields 722 for the preceding modes of scheduling. Finally, the days upon which the scheduling is operative can be set by clicking a check box in the area designated "Schedule for days" 724.

In order to support the construction of business processes, the data processing system should preferably provide the following atomic executable entities.

Firstly, Start Component, which defines the starting node for any graph comprising a plurality of executable entities. Usually, each business graph will comprise a single start node. However, embodiments of the present invention are not limited thereto and embodiments could equally well be realised in which several starting nodes are provided.

Secondly, it is preferable that there is provided an end component or node which provides a termination point for the execution of a business process. It will be appreciated that a business process may have several end nodes.

Thirdly, the User Selected Path 608 of figure 6 should enable the user of the system to specify a plurality of possible execution paths. When this executable entity is encountered a dialogue box, described below in relation to figure 8, is displayed which prompts the user to select the preferred path for continued execution. Finally, a second type of User

Selected Path should, preferably be available, in which the user can select parallel execution of selectable ones of a plurality of possible execution paths. Figure 8 depicts such a dialogue box 800 enabling the user to select links for determining the continued flow of execution. The user is presented with a plurality of node names 802, namely, User selection 1, User Selection 2 and User Selection 3. In order to govern which path is selected, the user selects one of the nodes presented in the Node Names text box 804 using a mouse as is known in the art. Execution of the business process continues according to the selected path.

The above illustrative embodiments have referred to the lack of a need to further modify said executable entities after selection thereof for execution. Modification includes, without limitation, at least one of either changing the means of exchanging data between executable entities and/or changing the methods which govern the behaviour of the executable entities.

Although the parameter area has been described or implemented in terms of a Visual Basic function, the present invention is not limited thereto. An embodiment of the present invention could also be realised using, for example, a globally declared array, a globally accessible linked list or merely pointers to globally accessible structures, defined according to a type definition, having an appropriately allocated memory block of memory for holding an identifier and a corresponding value associated with that identifier.

The reference signs in the description and claims are not intended to be limiting in anyway and are used only for the purpose of illustration.

The reader's attention is directed to all papers and documents which are filed concurrently with or previous to this specification in connection with this application and which are open to public inspection with this specification, and the contents of all such papers and documents are incorporated herein by reference.

All of the features disclosed in this specification (including any accompanying claims, abstract and drawings), and/or all of the steps of any method or process so disclosed, may be combined in any combination, except combinations where at least some of such features and/or steps are mutually exclusive.

Each feature disclosed in this specification (including any accompanying claims, abstract and drawings), may be replaced by alternative features serving the same, equivalent or similar purpose, unless expressly stated otherwise. Thus, unless expressly stated otherwise, each feature disclosed is one example only of a generic series of equivalent or similar features.

The invention is not restricted to the details of the foregoing embodiments. The invention extends to any novel one, or any novel combination, of the features disclosed in this specification (including any accompanying claims, abstract and drawings), or to any novel one, or any novel combination, of the steps of any method or process so disclosed.

What is claimed is:

CLAIMS

1. A method for processing data in or configuring a data processing system, said data processing system comprising memory for storing a plurality of executable entities, means for establishing an order of execution of selectable ones of the executable entities, and means for executing the selected executable entities according to said order, without modification of said selected executable entities, the method comprising the steps of

storing, in said memory, a plurality of executable entities,

establishing an order of execution of selectable ones of said executable entities, and

executing the selectable ones of said executable entities according to said order without modification of said executable entities.

2. A method as claimed in claim 1, further comprising the steps of
establishing, within said memory, a data structure, commonly accessible by said selected executable entities, which is arranged to allow said selected executable entities to store therein and retrieve

therefrom any data object and/or to allow indirect exchange of data between said selected executable entities.

3. A method as claimed in any preceding claim, wherein said data processing system further comprises an execution engine which instigates the execution of plurality of executable entities, and wherein the step of executing the executable entities comprises the steps of
providing each of said plurality of executable entities with a common interface for instigating execution of the executable entities, and
instigating, by said execution engine, execution of said two executable entities utilising said common interface.

4. A method as claimed in claim 3, wherein said executable entities have associated therewith data and a method for instigating execution thereof, the step of instigating execution further comprising the steps of
obtaining said data for instigating execution of the executable entity, and
executing said method.

5. A method as claimed in claim 4, wherein each executable entity, upon termination, produces one of

plurality of data values representative of the result of the execution of said entity, and said method further comprising the steps of

determining, in response to said data values representative of the result of the execution of said entity, an executable entity associated with said values, and

instigating execution of said executable entity associated with said values.

6. A method as claimed in claim any preceding claim, data processing system further comprises a clock, said method further comprising the steps of

associating a predeterminable execution time with at least one executable entity,

scheduling execution of said at least one executable entity, and

instigating execution of said at least one executable entity at the associated predeterminable execution time,

thereby delaying execution of said executable entity for or until a predeterminable period.

7. A method as claimed in claim 6, wherein the step of scheduling comprises at least one of the steps of

storing, in a data structure, an identifier of an executable entity and an associated time of execution thereof, and

repeatedly polling or checking said data structure to determine whether or not a said scheduled time has passed, and
instigating, in response to said repeated polling, execution of an executable entity having an associated execution time which has passed.

8. A method as claimed in any preceding claim, wherein said executable entity is at least one of an object, a function, a self-contained program, or a DLL task.

9. A data processing system comprising memory for storing a plurality of executable entities, means for establishing an order of execution of selectable ones of the executable entities, and means for executing the selected executable entities according to said order without modification of said selected executable entities.

10. A system as claimed in claim 9, further comprising means for establishing, within said memory, a data structure, commonly accessible by said selected executable entities, which is arranged to allow said selected executable entities to store therein and retrieve therefrom any data object and/or to allow indirect exchange of data between said selected executable entities.

11. A system as claimed in either of claims 9 to 10, wherein said data processing system further comprises an execution engine which instigates the execution of plurality of executable entities, and wherein means for executing the two executable entities comprises

means for providing each of said plurality of executable entities with a common interface for instigating execution of the executable entities, and

means for instigating execution of said executable entities utilising said common interface.

12. A system as claimed in claim 11, wherein said executable entities have associated therewith data and a method for instigating execution thereof, the means for instigating execution further comprising means for obtaining said data for instigating execution of the executable entity, and

means for executing said method.

13. A system as claimed in claim 12, wherein each executable entity, upon termination, produces one of plurality of data values representative of the result of the execution of said entity, and said data processing system further comprises

means for determining, in response to said data values representative of the result of the execution

of said entity, an executable entity associated with said data values, and
means for instigating execution of said executable entity associated with said data values.

14. A system as claimed in claim any of claims 9 to 13, wherein data processing system further comprises a clock,
means for associating a predeterminable execution time with at least one executable entity,
means for scheduling execution of said at least one executable entity, and
means for instigating execution of said at least one executable entity at the associated predeterminable execution time.

15. A system as claimed in claim 14, wherein means for scheduling comprises at least
means for storing, in a data structure, an identifier of an executable entity and an associated time of execution thereof, and
means for repeatedly polling or checking said table to determine whether or not a scheduled time has passed, and
means for instigating, in response to said repeated polling, execution of an executable entity having an associated execution time which has passed; or

means for generating an interrupt to the execution engine to indicate that execution of an associated executable entity should commence, and

means for instigating, in response to said interrupt, said executable entity.

16. A system as claimed in any of claims 9 to 15, wherein said executable is at least one of an object, a function, a self-contained program, or a DLL task.

17. A computer program product for processing data in or configuring a data processing system, said data processing system comprising memory for storing a plurality of executable entities, means for establishing an order of execution of selectable ones of the executable entities, and means for executing the selected executable entities according to said order, without modification of said selected executable entities, the computer program product comprising

computer readable program code means for storing a plurality of executable entities,

computer readable program code means for establishing an order of execution of selectable ones of said executable entities, and computer readable program code means for executing the selectable ones of said executable entities according to said order without modification of said executable entities.

18. A computer program product as claimed in claim 17, further comprising
computer readable program code means for establishing, within said memory, a data structure, commonly accessible by said selected executable entities, which is arranged to allow said selected executable entities to store therein and retrieve therefrom any data object and/or to allow indirect exchange data between said selected executable entities.

19. A computer program product as claimed in any of claims 17 to 18, wherein said data processing system further comprises an execution engine which instigates the execution of plurality of executable entities, and wherein computer readable program code means for executing the executable entities comprises
computer readable program code means for providing each of said plurality of executable entities with a common interface for instigating execution of the executable entities, and

computer readable program code means for instigating execution of said executable entities utilising said common interface.

20. A computer program product as claimed in claim 19, wherein said executable entities have associated therewith data and a method for instigating execution thereof, the computer readable program code means for instigating execution further comprises

computer readable program code means for obtaining said data for instigating execution of an executable entity, and

computer readable program code means for executing said method.

21. A computer program product as claimed in claim 20, wherein each executable entity, upon termination, produces one of plurality of data values representative of the result of the execution of said entity, and said computer program product further comprises

computer readable program code means for determining, in response to said data values representative of the result of the execution of said entity, an executable entity associated with said data values, and computer readable program code means for instigating execution of said executable entity associated with said data values.

22. A computer program product as claimed in any of claims 17 to 21, wherein the data processing system further comprises a clock, said computer program product further comprises
computer readable program code means for associating a predeterminable execution time with at least one executable entity,
computer readable program code means for scheduling execution of said at least one executable entity, and
computer readable program code means for instigating execution of said at least one executable entity at the associated predeterminable execution time.

23. A computer program product as claimed in claim 22, wherein the computer readable program code means for scheduling comprises at least one of the
computer readable program code means for storing, in a data structure, an identifier of an executable entity and an associated time of execution thereof, and
computer readable program code means for repeatedly polling or checking said data structure to determine whether or not a scheduled time has passed, and

computer readable program code means for instigating, in response to said repeated polling, execution of an executable entity having an associated execution time which has passed.

24. A computer program product as claimed in any of claims 17 to 23, wherein said executable is at least one of an object, a function, a self-contained program, or a DLL task.

25. A data processing system or method as claimed in any preceding claim substantially as described herein with reference to and/or as illustrated in the accompanying drawings.

26. A data processing system and method substantially as described herein with reference to and/or as illustrated in the accompanying drawings.

27. A data processing system comprising a plurality of executable entities, a memory for storing the plurality of executable entities, a display means for producing a graphical representation of the plurality of executable entities, means for executing said executable entities, input means for selecting for execution selectable ones of said plurality of executable entities, the system comprising means for displaying on said display

means a graphical representation of said plurality of executable entities, means for selecting at least two of the executable entities for execution, means for establishing an order of execution for said at least two executable entities, and means for executing said at least two executable entities according to said order without modification of said at least two executable entities.

28. A method for configuring a data processing system, said data processing system comprising a plurality of executable entities, a memory for storing the plurality of executable entities, a display means for producing a graphical representation of the plurality of executable entities, means for executing said executable entities, input means for selecting for execution selectable ones of said plurality of executable entities, the method comprising the steps of displaying said plurality of executable entities, selecting two of the executable entities for execution, establishing an order of execution for the two executable entities, and executing the two executable entities according to said order without modification of said two executable entities.

The Patent Office

55

Application No:	GB 9625443.8	**Examiner:**	B G Western
Claims searched:	1-28	**Date of search:**	21 May 1997

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.O): G4A APL APX

Int Cl (Ed.6): G06F 9/44 9/46

Other:

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
X	GB-2266982-A (NISSAN) See whole document	1-28
X	GB-2077966-A (HITACHI) See whole document	1,2,8-10,16-18, 24,27,28
X	EP-0710909-A1 (BOSTON TECHNOLOGY) See whole document	1,2,8-10,16-18, 24,27,28
X	WO-88/07719-A2 (AIMTECH) N.b. pp1-26, Figs 8-12	1,2,8-10,16-18, 24,27,28
X	US-5386568-A (WOLD et al) See whole document	1,8,9,16,17, 24,27,28
X	WordPerfect 6.0a, 20 April 1994, WordPerfect Corporation See Help file re macros, OLE, DDE.	1,2,8-10,16-18, 24,27,28

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.